International Journal of Computer Networking,
Wireless and Mobile Communications (IJCNWMC)
ISSN (P): 2250-1568; ISSN (E): 2278-9448
Vol. 7, Issue 2, Jun 2017, 19-28

THROUGHPUT ANALYSIS OF TCP NEWRENO FOR MULTIPLE BOTTLENECKS

MOHIBUR RAHMAN[1], LUTFUN NAHAR[2], ZINNIA SULTANA[3] & RASHEDUL ISLAM[4]

[1]*Department of Computer Science & Engineering,*
University of Chittagong, Bangladesh
[2,4]*Lecturer, Department of Computer Science & Engineering,*
International Islamic University Chittagong, Bangladesh
[3]*Associate Professor, Department of Computer Science & Engineering,*
International Islamic University Chittagong, Bangladesh

ABSTRACT

A very complex problem in the network is –congestion control. It is one of the major problems among top listed network problems. A lot of important data might be àt loss, as a result, resources become wasted. Many TCP models have proposed to protect the loss during data transfer. They differ from each other on the basis of algorithm for congestion control and segment loss recovery technique. TCP NewReno is one of them that we have proposed in our paper, and it is also an analytical model for the throughput of TCP NewReno. This performance is given for multiple bottlenecks -a queue management mechanism. In our paper, we show that increased throughput of TCP New Reno over Reno and how NewReno suffers from less packet loss rate than the Reno model, by using the NS-2 simulator.

KEYWORDS: TCP, Reno, UDP & NewReno

Received: Jan 02, 2017; **Accepted:** Jan 24, 2017; **Published:** Jun 03, 2017; **Paper Id.:** IJCNWMCJUN20172

Original Article

INTRODUCTION

There are two unique transport-layer protocols, one is User Datagram Protocol (UDP) and the other is Transmission Control Protocol (TCP). An unreliable and connectionless protocol is UDP, where TCP is a connection-oriented, reliable protocol. In case of TCP, a connection must be established before any data, when the data transferred between two processes, a connection must be established before data transfer. IT gives reliable data transfer by using sequence numbers, flow control, timers, and acknowledgements. It guarantees that data is transferred from the sender to the receiver, correctly and in-sequence. r. It gives full-duplex, one way data transfer in the Internet. To prevent loss, it provides various mechanisms. Congestion control is one of them, that prevents a router from overflowing.

Literature Review

TCP, the congestion control mechanism is used to protect the network from overflowing of data packets. TCP uses end-to-end congestion control mechanism, since the IP layer provides no explicit feedback to the end systems regarding network congestions. It dynamically resizes the congestion window, depending on the behavior of the network- packet loss, delay, etc. It increases the congestion window size when there is available bandwidth and decreases the window when there is congestion. The TCP congestion control mechanism, at each side of a connection keep track of a variable, the congestion window (cwnd). It imposes the following constraint on the sending rate: LastByteSent - LastByteACKed≤ min (cwnd, rwnd) this indicates that, the amount of

unacknowledged data must not exceed the minimum of cwnd and round. Thus, the sender's sending rate is roughly the sound/RTT. The congestion control mechanisms use two events, as the indication of network congestion: timeout and triple duplicate ACKs. When the timer of a segment expires, it is considered that the segment is lost. Then, the segment is retransmitted. If the sender receives 3 duplicate acknowledgements for a segment, then the sender considers it as an indication of loss of the next contiguous segment. Then it retransmits the next segment. The congestion control mechanism has four algorithms to recover the lost data and utilize the network resources. These algorithms are: slow-start, congestion avoidance, fast-retransmit, and fast-recovery.

When the sender receives triple duplicate ACKs for a segment, it considers this event as congestion in the network. Then, it reduces its congestion window by halving the size of the window that was before the loss event occurs. Then, the sender increases its sending rate by increasing the window size by 1 MSS for every RTT. This linear increment phase is known as *congestion avoidance* [7], and the whole process is known as additive-increase, multiplicative-decrease. In this process the actual increment for the congestion window is: increment = MSS ・ (MSS/cwnd) cwnd += increment

Weekend grows by a fraction of MSS for every ACK, instead of an entire MSS. Here, sound decreases aggressively, but increases conservatively, because the consequences of having too large a window are much worse than those of it is too small. It is recommended to check that, the congestion window never drops below 1 MSS. At the beginning of a TCP connection the congestion window is initialized to 1 MSS that results a sending rate of MSS/RTT. It is not preferable to send data at this rate as the connection band width may be much larger than this rate. So it is required to increase the congestion window to utilize the bandwidth. In this phase, the sender grows its sending rate exponentially increasing the congestion window by 1 MSS for each acknowledgement, until there is a loss event. This mechanism is known as *slow start* [7], as the sender begins at a slow rate. The sender also enters into the slow-start phase after a coarse-grained timeout. When a timeout event occurs the sender considers it as the detection of packet loss. As the response to this event, the sender sets its congestion window to 1 MSS, that means it enters the slow-start. Then, it increases the window exponentially until it reaches one half of the value it had before the timeout, the threshold value- ssthresh. When it reaches that value, then it increases the window linearly, that is it enters the congestion avoidance phase. When a duplicate ACK is sent by the sender, it realizes that a packet out-of-order must have received from the other side, which indicate that an earlier packet might have been lost. And, it is also possible that the earlier packet has been delayed rather than lost, the sender waits for three duplicate ACK to ensure a loss occurs. After receiving a triple duplicate ACK the sender immediately transfers the missing segment after the segment's timer expires, then it enters into the additive-increase phase. This process is known as *fast retransmit* [7]. This technique is able to remove about half of the coarse-grained timeouts on a typical TCP connection that improves the throughput about 20%. When the fast-retransmit mechanism signals congestion, rather than lost the congestion window all the way back to one packet and execute slow-start. This mechanism, known as fast recovery [7], effectively eliminate the slow-start phase that happens between when fast retransmit detects a lost packet and additive increase begins.

VARIANTS OF TCP

Depending on the use of the congestion control mechanisms described above, TCP introduces several variants- Reno [3], NewReno [1], Vegas [4], Tahoe [2], the SACK [5]. In this section, some description of these variables is given. TCP Tahoe [2] uses slow-start and congestion-avoidance process as the congestion control mechanism. Tahoe [2] detects packet losses only by timeouts. In Tahoe [2] whenever a TCP connection detects a packet loss by timeout, it should go

through the slow-start. It happens because, Tahoe [2] considers the loss event as an initial burst that might overwhelm the network and the connection might never get started. For congestion avoidance, it uses additive increase multiplicative decrease. A sign of congestion is packet losing and Tahoe saves the half of the current window as a threshold value. It then sets count to one and begins slow start until it reaches at the threshold value. After that, it increments linearly until it encounters a packet loss. The problem with this TCP variation is that it takes a complete timeout interval to detect a packet loss that offers a major cost in high bandwidth-delay product links. A class of models that can be estimated using pool estimation, can be written as The TCP SACK [5] is an extension of TCP Reno [3]. It retains the slow-start and fast-retransmit parts of Reno [3]. Here the receiver uses selective acknowledgement instead of cumulative, that acknowledges out-of-order data. It uses the slow-start and fast-retransmit as like as Reno [3] along with the coarse grained timeout behavior of Tahoe [2] to fall back on when by the modified algorithm a packet loss is not detected. Whenever the sender enters the fast recovery, it initializes a variable pipe, an estimate of how much data is outstanding in the network, and it also sets sound to half of the current size. For the reception of an ACK, it reduces the pipe by 1 and for each retransmission it increases it by 1. Whenever the pipe goes smaller than the cwnd checks, which segments are unacknowledged and send them. If there are no outstanding segments, then it sends a new packet. The biggest problem with SACK [5] is that, it requires selective acknowledgements which is difficult to implement. Here, in this paper, we discussed about the TCP NewReno [1] which is a slight modification of the TCP Reno [3]. It has the capability to detect multiple packet losses and also more efficient than Reno [3] in the event of multiple packet losses. It uses slow-start, congestion-avoidance algorithm, and fast-retransmit as like as Reno [3], and also recovers from multiple losses by using an advanced fat-recovery algorithm in a single window, avoiding many of the retransmissions that occur in the Reno [3].

ANALYSIS OF PROPOSED MODEL

The analytical model is designed in two steps – 1) model without timeout (No TO) and 2) full mode. It uses following notations –

Table 1: Parameter Definition of Model

Parameter	Parameter Definition
p	Loss event rate
q	Segment loss rate within a loss event
R	Average round-trip time
RTO	Average duration of first timeout in a series of timeouts
W	Average of the peak congestion window size

MODEL WITHOUT TIMEOUT/NOTO MODEL

In this section, it is mentioned that all loss rates are determined only by triple duplicate ACKs, no timeout occurs. In our proposed method we consider that each cycle has a congestion avoidance period (CAFR). It is shown that here, two adjacent CAFR is unique. Suppose the expected number of segments between the first loss and the last loss is δ. Now, for m uniformly spaced drops in a typical window of size W the expected number of segments between the first loss and the last loss can be expressed as,

$$\delta = W\text{-}WE[1/m]=W\text{-}W/E[m] \quad (1)$$

As in the drop window, the first segment is always lost, then the probability of m-1losses from the remaining W-1 segments follows the binomial probability mass function. Now, the probability of m segment losses from a drop window of size W can be written

A(W,m) $=C_{m-1}^{W-1}(1-q)^{W-m}q^{m-1}$ Where

C_{m-1}^{W-1}Represents the binomial coefficient.

Since it is assumed that all losses are identifiable by triple duplicate ACKs andm≤W-3. Now, the expected value of m,

$$E[m]=\sum_{m=1}^{W-3} mA(W,m) =\sum_{m=1}^{W-3} mC_{m-1}^{W-1}(1-q)^{W-m}q^{m-1}) = W(W+1)^2(1-q)^{W-1}2^{W-2} =1+(W-1)q=1+Wq \quad (2)$$

Substituting E [m] into equation (1) it is found that:

$\delta=W^2q/1+Wq$. By using δ we find the expected number of transmitting segments found at:

$S_{CAFR}=1/p+(W^2q/1+Wq)$ (3)

Again, to compute W in terms of p and q, that can be expressed as, the sum of the expected number of segments in the linear increase phase, $S_{LI,}$ the expected number of segments transmitted from the start of round W/2+2 until triple exterminate congestion avoidance s_β, and the expected number of segments transmitted during fast recovery, S_{FR}. Therefore,

$SCAFR=S_{LI}+S_\beta+S_{FR}$ (4)

When TCP detects a segment loss and enters fast recovery, the expected number of outstanding segments is W. For m drops, where m≤W/2, the source receives a W-duplicate ACKs, each of which increases the congestion window by 1 segment. So in the first RTT the source sends W/2-m new segments. After receiving the 1st partial ACK the second RTT starts. During this time TCP transmits W/2-m+1 new segment. So, for mlosses fast recovery requires m RTTs that transmit W/2-m+j-1 new segments during jthRTT. So, it is obtained that:

$$S_{FR}^{m\le W/2}=\sum_{j=1}^{m}\frac{W}{2}-m+j-1 = (\frac{W}{2}-m)+(\frac{W}{2}-m+1)+(\frac{W}{2}-m+2)+\ldots.=\frac{Wm}{2}-m^2+\frac{m(m-1)}{2}=\frac{m}{2}(W-m-1) \quad (5)$$

But for m>W/2, no new segment is transmitted. For one PA the congestion window is increased only by one segment. It requires (m-W/2+1) partial ACK to transmit a new segment. Then

$$S_{FR}^{m>W/2}=\sum_{k=m-\frac{W}{2}+1}^{m-1}\frac{W}{2}-m+k \quad =\sum_{k=1}^{m}\left(\frac{W}{2}-m+k\right)-\left[\sum_{k=1}^{m-\frac{W}{2}}\left(\frac{W}{2}-m+k\right)+W/2\right]$$

$$=\left[\frac{Wm}{2}-m^2+\frac{m(m-1)}{2}\right]-\left(\frac{W}{2}-m\right)\left(m-\frac{W}{2}\right)+\left]\frac{\left(m-\frac{W}{2}\right)\left(m-\frac{W}{2}+1\right)}{2}+\frac{W}{2}\right.$$

$$=\frac{W^2}{8}-\frac{W}{4} \quad (6)$$

Now using equation (5) and equation (6) the expected number of transmitting segments during fast-recovery can be found at,

$$S_{FR}=\sum_{m=1}^{W/2}A(W,m)S_{FR}^{m\le\frac{W}{2}}+\sum_{m=\frac{W}{2}+1}^{W-3}A(W,m)S_{FR}^{m>\frac{W}{2}}=\frac{W^2}{2}(q-q^2)+\frac{W}{2}(1-5q+3q^2)-(1-2q+q^2) \quad (7)$$

After receiving a full ACK, fast-recovery is terminated. During this time, the congestion window size is W/2 and it increases by one segment by per round until the peak value of W in round W/2+1 reaches. Then, the expected number of transmitting segment is,

$$S_{LI} = \sum_{i=W/2}^{W} i = \sum_{i=1}^{W} i - \sum_{i=1}^{\frac{W}{2}-1} i = \frac{W(W+1)}{2} - \frac{\left(\frac{W}{2}\right)\left(\frac{W}{2}-1\right)}{2}$$

$$= \frac{3W^2}{8} + \frac{3W}{4} \quad (8)$$

Now, for the segments transmitted between a loss occurs and the first lossdetected, two cases are found-

- $S\beta=0$, if the first loss occurs at the start of round W/2+1.
- S_β=W-1, if the first loss occurs at the end of round W/2+1.

Therefore, it can be approximated that,

$$S_\beta = W/2 \quad (9)$$

Substituting the values from equations (7), (8), (9) in the equation (4) it is found that,

$$S_{CAFR} = \{\frac{W^2}{2}(q-q^2)+\frac{W}{2}(1-5q+3q^2)-(1-2q+q^2)\}+(\frac{3W^2}{8}+\frac{3W}{4})+\frac{W}{2}$$

$$= (\frac{3}{8}+\frac{q}{2}-\frac{q^2}{2})w^2+(\frac{7}{4}+\frac{5q}{2}+\frac{3q^2}{2})W-(1-2q+q^2) \quad (10)$$

Equating the right hand side of equation (3) and equation (10) the value ofcongestion window size W is found as, $\frac{10pq-5p+\sqrt{p(24+32q+49P)}}{P(3+4q)}$ (11)

The expected time duration of a CAFR period can be expressed as,

$$D_{CAFR} = D_{FR} + D_{LI} + D_\beta \quad (12)$$

Where, the duration of a linear increase period is D_{LI}, the expected delay from the start ofround (W/2+2) to the end of congestion avoidance is D_β, and is the fast-recoverydelay $D_{FR.}$

Now, the duration of the linear increase period is

$$D_{LI} = (\frac{W}{2}+1)R \quad (13)$$

As the fast recovery requires m RTTs form segment losses, so the duration of thefast recovery would be,

$$D_{FR} = (1+Wq)R \quad (14)$$

It is approximated that the duration from the start of round (W/2+2) to the end of congestion avoidance would be,

$$D_\beta = R/2 \quad (15)$$

Substituting the values from the equation (13), (14) and (15) into equation (12) the duration is found as,

$$D_{CAFR} = \left(\frac{W}{2}+1\right)R + R/2+(1+Wq)R$$

$$= \left(\frac{W}{2}+Wq+\frac{5}{2}\right)R \quad (16)$$

Substituting equation (3) and (16) the throughput can be written as,

$$T(NoTO) = \frac{\frac{1}{p}+\frac{W^2q}{1+Wq}}{\left(\frac{W}{2}+Wq+5/2\right)R} \quad (17)$$

FULL MODEL

In this model, besides fast recovery and congestion avoidance phase, timeout is considered. The fast recovery starts when there are no more than W-3 segments lost. During this time, if any retransmitted segments lost a timeout will occur. NewReno requires mRTTs for my segment losses in a drop window, sending W/2 segments per RTT including retransmission. Then, the probability of timeout can be expressed as,

$$P_{IFR}=\sum_{m=1}^{W-3} A(W,m)[p+(1-p)p+\cdots+(1-p)^{\frac{mW}{2}-1}p]$$

$$=\sum_{m=1}^{W-3} A(W,m)[1-(1-p)^{\frac{mW}{2}}] \quad (18)$$

By using equation 18. It can be expressed that

$$P_{TO}=\sum_{m=W-2}^{W} A(W,m)+\sum_{m=1}^{W-3} A(W,m)[1-(1-p)^{\frac{mW}{2}}]$$

$$=[\sum_{m=1}^{W} A(W,m)-\sum_{m=1}^{W-3} A(W,m)]+\sum_{m=1}^{W-3} A(W,m)[1-(1-p)^{\frac{mW}{2}}]=1-\sum_{m=1}^{W-3} A(W,m)[(1-p)^{\frac{mW}{2}}] \quad (19)$$

During timeout, no new segment is transmitted, that is,

P_{TO}= 0and the duration would be,

$$D_{TO}=RTO\frac{1+p+2p^2+4p^3+8p^4+16p^5+32p^6}{1-p} \quad (20)$$

In the slow start phase, the congestion window is doubled per RTT. This phase continues until the slow start threshold reached which is the half of the congestion window transmitted before the loss event. Then, TCP enters the congestion avoidancephase. Hence, the transmitted segment during this phase,

$$S_{SS}=1+2+4+\ldots+W/4=2^{1+log\frac{W}{4}}-1 \quad (21)$$

And the duration would be

$$D_{SS}=(\log\frac{W}{4}+1)R \quad (22)$$

Therefore, the throughput for the full model can be obtained by following the approach in [3] as,

$$T_{FULL}=\frac{\frac{1}{p}+\frac{W^2q}{1+Wq}}{NR+P_{TO}(1+2P+4P^2)R+(1+\log(W/4)R)} \quad (23)$$

SIMULATION RESULTS

In this segment, we illustrate TCP NewReno performs how much better thanTCP Reno, using NS-2 simulator. Here, experiments have two long duration flows: one NewReno flow and one Reno flow. Here, three TCP sources and one UDP source are used. The bottleneck band widths vary from 0.7Mbps to 2Mbps. Here, two types of traffic flows are used-FTP and constant bit rate UDP flows. The propagation delays vary from 1ms to 20ms.Each session consists of a unique

client/server pair. The waiting time of a client between the reception of a file and issuing the next request is exponentially distributed and has a mean of 0.3ms. The file sizes are drawn from a Pareto distribution with mean of 10KB and shape 1.5. The maximum congestion window size is 8KB and packet size is 0.5KB. The UDP flow uses a rate of 0.01Mbps, the packet size is 1KB, and the propagation delay is 7ms. The FTP flows start at uniformly distributed times between 0 and 7 seconds.

NETWORK MODEL AND TRAFFIC MODEL

The simulation is done here for multiple bottleneck-link.

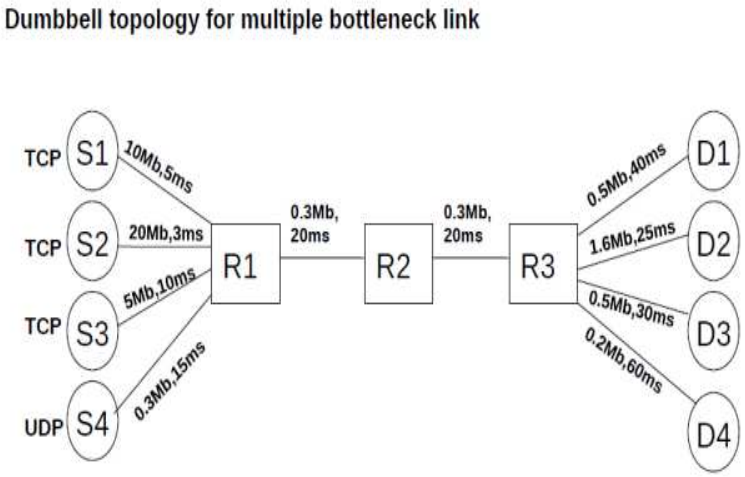

Figure 1: Multiple Bottleneck Link

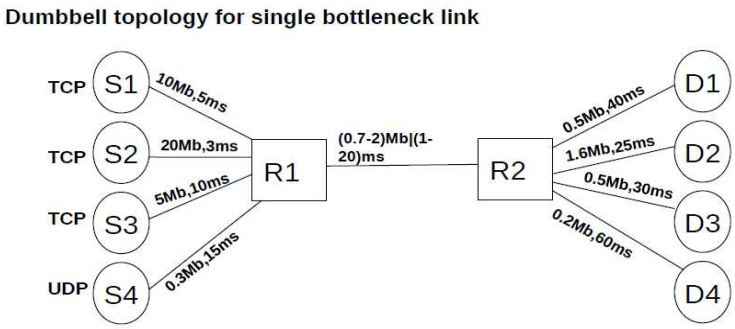

Figure 2: Single Bottleneck Link

MULTIPLE BOTTLENECK

Now, the experiment is conducted with two bottleneck links connected in series. Each link has a bandwidth of 0.3Mbps, propagation delay 20ms, and the buffer size is 10 packets. The congestion window size is 8KB and packet size is552bytes. The throughput change for each model with time is shown in Figure 3.

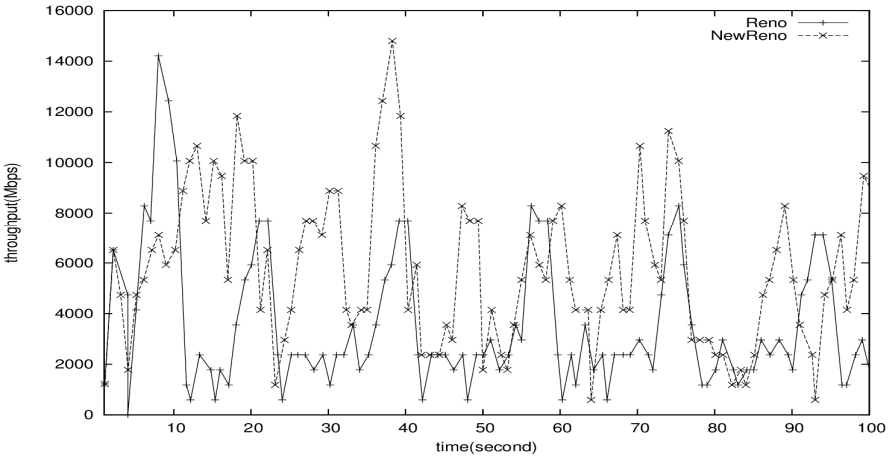

Figure 3: Throughput Change for Each Model

In case of multiple bottlenecks, TCP NewReno provides 10%-11% more throughput than TCP Reno. In Figure 4 loss rates for these two models are shown-

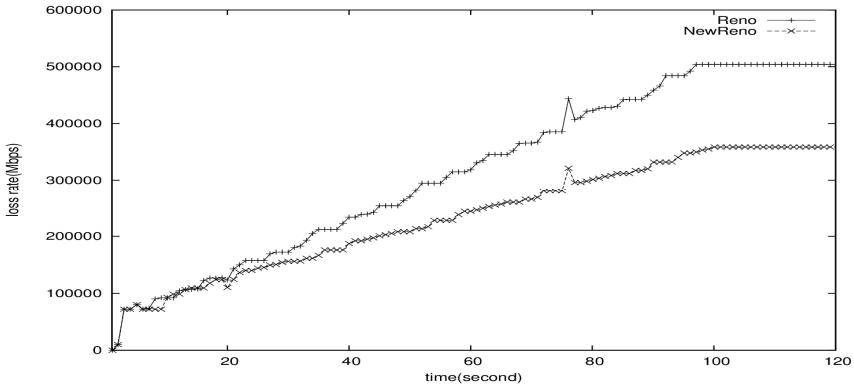

Figure 4: Loss Rates for Two Model

From the simulation results, it is found that Reno suffers from 20%-25% losses than NewReno. Thus, the simulation results indicate that TCP NewReno is more advantageous than TCP Reno, as the latter one suffers from more losses and gives lower throughput between the two.

CONCLUSIONS

In this paper, we describe the Mathematical analysis of the two model analytical models without timeout/No TO model and FULL model. The performance is evaluated in case of multiple bottleneck links. From the simulation results, we found TCP NewRenogives better performance than the Reno model in both throughput and for a packet losing rate that is better throughput with less packet losing rate. As a result, the congestion avoidance mechanism of TCP NewReno is more efficient than Reno. The significant performance of NewReno advantages over Reno.

REFERENCES

1. *NadimParvez, AnirbanMahanti, and Carey Williamson, "An Analytic ThroughputModel for TCP NewReno,"IEEE/ACM Transactions on Networking, Vol. 18, No.2, April 2010.*
2. *V. Jacobson,"Congestion Avoidance and Control," In Proc. of ACM SIGCOMM, pp. 314-329, August 1988*
3. *V. Jacobson,"Berkeley TCP evolution from 4.3-Tahoe to 4.3 Reno," In Proc. Of the8th IETF, Canada, August-1990.*
4. *L. Brakmo, S. O' Malley, and L. Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance," In Proc. of ACM SIGCOMM, pp. 24–35, New York, USA, August 1994.*
5. *K. Fall and S. Floyd,"Simulation-Based Comparisons of Tahoe, Reno, and SackTCP. ACM Computer Communication Review," 26(3):5–21, July 1996.*
6. *C. Samios and M. Vernon,"Modeling the Throughput of TCP Vegas," In Proc. OfACM SIGMETRICS, San Diego, USA, June 2003.*
7. *A. Misra and T. Ott,"The Window Distribution for Idealized TCP Congestion Avoidance with Variable Packet Loss," In Proc. of IEEE INFOCOM, pp. 1564–1572, New York, USA, March 1999.*
8. *J. Padhye, V. Firioiu, D. Towsley, and J. Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation," In Proc. of ACM SIGCOMM, Vancouver, Canada, September 1998.*
9. *Pathy, Kumar Chittarajan. D.R. (Ed). 2007. Forest, Government, and the Tribe. New Delhi: Concept of Publishing Company.*
10. *Chaudhuri, Buddhadeb. (2007).Forest And Tribals A historical Review of Forest Policy, Forest. In Dr. Chittarajan Kumar Pathy (Ed.), Forest, Government and the Tribe, (pp.1-17). New Delhi: Concept of Publishing Company*
11. *Mullick, Bosu Sanjay. (2007).State Forest Policy and Adivasi Self Rule in Jharkhand. In Dr. Chittarajan Kumar Pathy (Ed.), Forest, Government and the Tribe, (pp.2-29). New Delhi: Concept of Publishing Company*